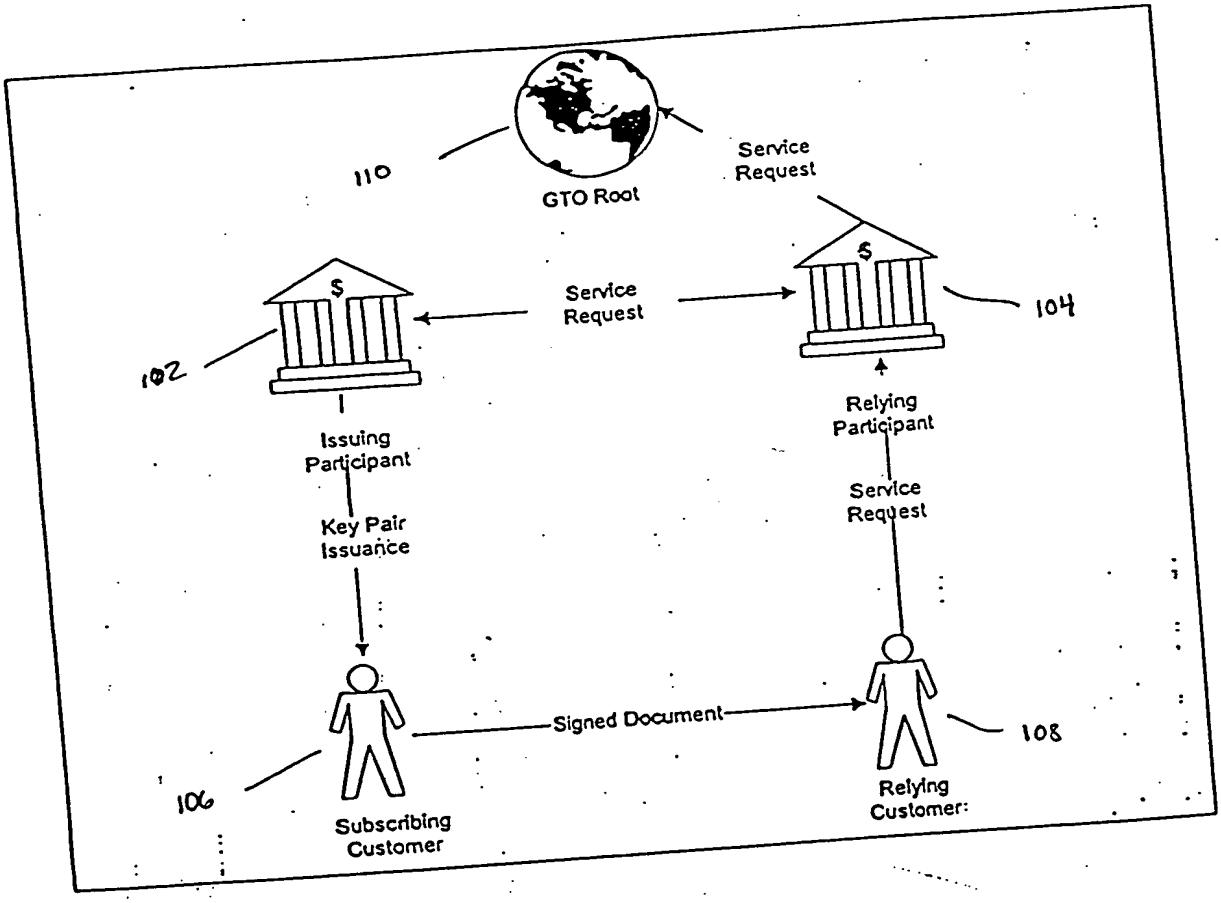
110
GTO Root
Service Request
Service Request
102
104
Issuing Participant
Relying Participant
Key Pair Issuance
Service Request
Signed Document
106
108
Subscribing Customer
Relying Customer

FIG. 1

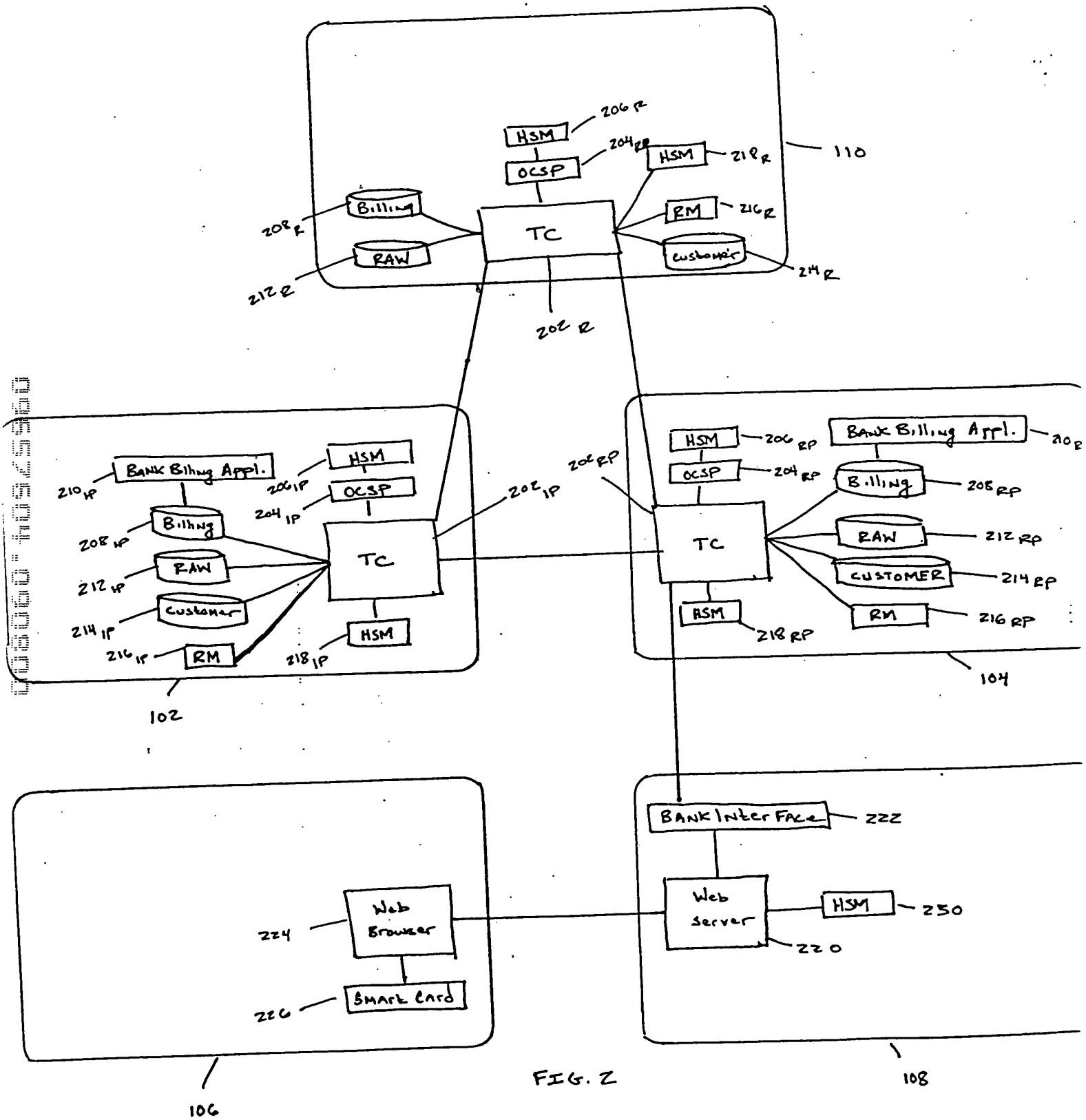
HSM
206R
OCSP
204RP
HSM
218R
110
Billing
208R
TC
RM
216R
RAW
212R
Customer
214R
202R
HSM
206IP
Bank Billing Appl.
210IP
OCSP
204IP
202IP
Billing
208IP
TC
RAW
212IP
Customer
214IP
RM
216IP
HSM
218IP
102
HSM
206RP
Bank Billing Appl.
210R
202RP
OCSP
204RP
Billing
208RP
TC
RAW
212RP
CUSTOMER
214RP
RM
216RP
ASM
218RP
104
Bank Interface
222
Web Browser
224
Smart Card
226
Web Server
HSM
250
220
106
108

FIG. 2

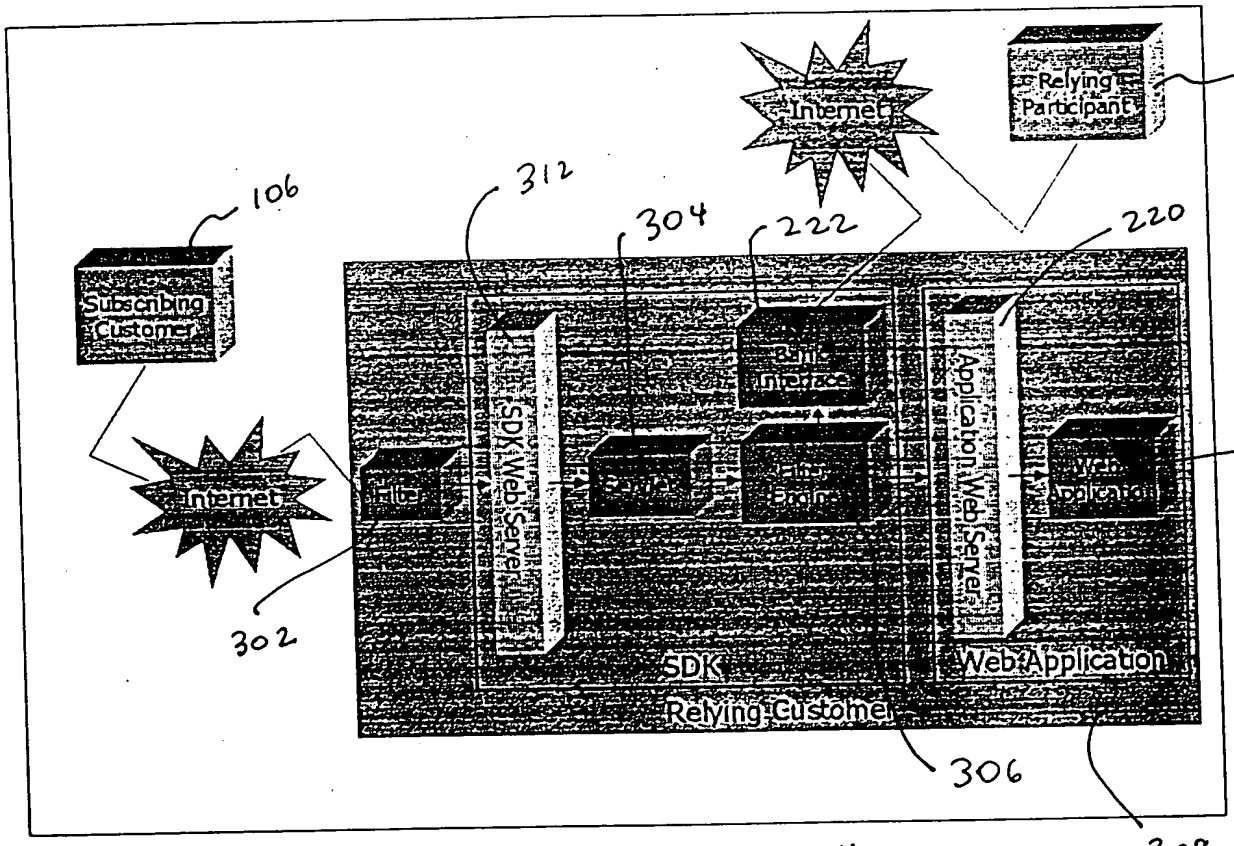


Figure 3 DSMS Passive Integration

```
// Redirect anything specific to an application to go through the
// DSMS

DWORD CDSMSIapiFilterFilter::ONPreprocHeaders(CHttpFilterContext* pCtxt,
        PHTTP_FILTER_PREPROC_HEADERS pHeaderInfo)
{
        // TODO: React to this notification accordingly and
        // return the appropriate status code

        // Filter looks for the pattern "/avf" and replaces the URL
        // with "/Servlet/DSMS_Servlet" (Will be modifiable using INI file)
        static const char strAppPath[] = "/avf";
        static const char strServletPath[] = "/Servlet/DSMS_Servlet";

        char buffer[1024], *newURL;
        DWORD buffersize = sizeof(buffer);
        LVOID urlBuffer = pCtxt->AllocMem(1024);

        newURL = (char*) malloc(1024);

        pHeaderInfo->GetHeader (pCtxt->m_pFC, "url", buffer, &buffersize);
        char *pPos = strstr(buffer, strAppPath);

        // the URL should not include http://<server name>
        if (pPos) {
                strcpy( newURL, strServletPath );
                strcat( newURL, pPos);

                pHeaderInfo->SetHeader (pCtxt->m_pFC, "url", newURL);
        }

        return SF_STATUS_REQ_NEXT_NOTIFICATION;
}
```

Fig. 4

| Steps |
| --- |
| Load Servlet properties from the properties file |

| |
| --- |
| Read data from the HTTP request |
| Create a hash table (name, value pairs) with parameters for the Filter Engine including HTTP headers, Content type, client IP address, HTTP method (GET and SET) and the actual data in the request |
| Identify if the data has been signed. If not signed, call Filter Engine with the hash table |
| If signed, URL decode the PKCS#7 message received from the Plug-In and insert it into the hash table |
| Call the Filter Engine with the hash table |
| Process the return value from the Filter Engine |
| If the return value from the Filter Engine indicates that the web application has been called, then display the next page |
| If the return value from the Filter Engine indicates that the page needs to be signed, the state of the Filter Engine is stored in a cookie and the page with the Plug-In is displayed |
| If the return value from the Filter Engine indicates that the Client Certificate is GOOD, then change the State and send a request to Filter Engine to retrieve the next page. |
| For all other values or exceptions, display error page to the client. |

Fig. 5

```
#Sign request when user adds a new document to AltaVista
BIModeSync; BISCertStatCheck; cookie, AltaVistaForum_AuthToken, dsmith;
url, newDocForm

#Sign request when user modifies a document in AltaVista
BIModeSync; BISCertStatCheck; cookie, AltaVistaForum_AuthToken, dsmith;
URL, modDocform

#Sign request when user deletes a document from AltaVista
BIModeSync; BISCertStatCheck; Cookie, AltaVistaForum_AuthToken,
dsmith; url, delItemsForm

#Test
#BIModeSync; BISCertStatCheck;Iname,identrus;city,Boston
```

Fig. 6

```
package com.identrus.filterengine;

import java.rmi.*;
import java.util.*;
import com.identrus.util.*;

/**
This is an interface definition for the Filter Engine Server
*/

public interface IRmiFEServer extends Remote{
public ReturnObject Service (Hashtable table)
throws RemoteException;
}
```

Fig. 7

Fig. 8

| Filter Engine Startup Steps |
|---|
| Loads Filter Engine properties from the properties file |
| Open log files |
| Load SSL or Utility Certificates |
| Load RMI server Policy File |
| Load Rules files into the memory |
| Validate Rules to verify correct formatting |
| The Filter Engine Interface is now ready to receive requests |

| Filter Engine Processing Steps |
|---|
| Receives HTTP Request data and the State from the Servlet |
| If the State passed from the Servlet is FE_NEW_REQUEST, the Filter Engine compares the request against the signing rules and determines whether the request has to be signed or not. It builds the Return Object specified in the FE_NEW_REQUEST State. |
| If the State passed in from the Servlet is FE_SIGNED_DATA, then it calls the Bank Interface to check the status of the Certificate. After interacting with the Identrus network, the Bank Interface returns the status. The status and the data in the CMS message are put into a Return Object and sent to the Servlet |
| If the State passed from the Servlet is FE_REQUEST_CHECKED, indicating the final stage of a signed transaction, the Web Application is called. The original page is retrieved from the Web Application and its content is returned to the Servlet in a Return Object |
| Log all signed request to the event log and all errors to the error log |
| All exceptions are returned to the Servlet as a part of the Return Object |

Fig. 9

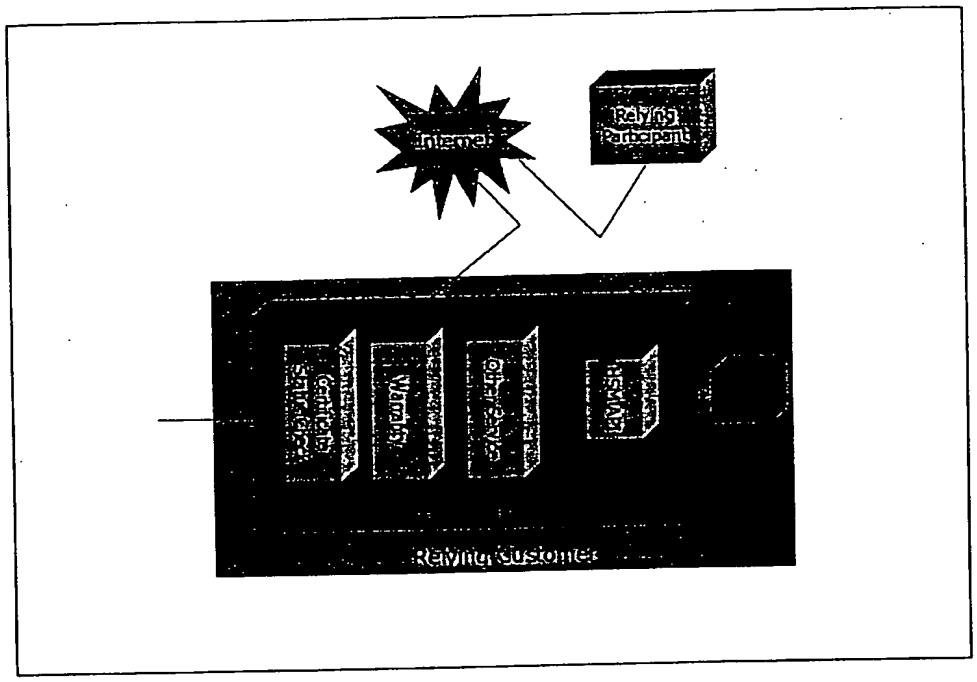


Fig. 10

| Bank Interface Startup Steps |
| --- |
| Loads Bank Interface properties from the properties file |
| Open log files |
| Load SSL or Utility Certificates |
| Load RMI server Policy File |
| Load cryptographic modules, either software or hardware (Hardware Security Module API) as specified in the properties file |
| At this stage the Bank Interface is ready to receive service request |
| Call Bank Interface service manager with the DSMS request that contains the name of the service, mode of the service and the message |

Fig. 11

| Steps |
| --- |
| Retrieve Relying Customer and Root Certificate from the server |
| Retrieve Subscribing Customer and Issuing Participant's Certificate from the CMS (Cryptographic Message Syntax) also referred as PKCS#7. |
| Verify signature on the CMS message |
| Verify signature on the Subscribing Customer's Certificate using the Issuing Participant's Certificate |
| Verify signature on the Issuing Participant's Certificate using the Identrus Root Certificate that belongs to the Relying Participant |
| The Validity period is then checked on the two Certificates received against the current date |
| Retrieve the OCSP responder's URL from the Relying Customer's certificate |
| Create an OCSP request for the Subscribing Customer's Certificate signed by the Relying Customer. All OCSP requests contain a Service Locator Extension, which is set by the Authority Information Access (AIA) extension defined in the certificate |
| Log the OCSP request to the transaction log |
| Create HTTP(S) connection to the OCSP responder and send the OCSP request. |
| Receive OCSP response from the responder and verify the signature using the OCSP Responder's Certificate |
| Get the status of the certificate from the Response |
| Repeat steps 8 through 11 for the Issuing Participant and the Relying Participant's OCSP Responder's certificate |
| Log the OCSP response to the transaction log |
| If the status of all the responses are GOOD return GOOD, else return the status |
| Log all signed request to the event log and all errors to the error log |
| All exceptions are returned to the client as a part of the Return Object |

Fig. 12

| # | Description | Protocol |
|---|---|---|
| 1 | User clicks 'Submit' button on HTML Form in Web Browser | HTML UI |
| 2 | Web Browser posts form data to SDK Web Server | HTTP |
| 3 | SDK Web Server passes all requests to Servlet. | |
| 4 | Servlet passes request to Filter Engine. | RMI |
| 5 | Filter Engine creates a Return-to-Browser URL (as a GET with parameters for data) representing the data of the original POST or GET form posting and returns it along with instructions to get the data signed to the Servlet | RMI |
| 6 | Servlet builds a response with<br>1. An Applet tag pointing to the Client Interface Applet OR<br>2. A call to a browser plug-in and the arguments Return-to-Browser URL and the data to sign. | Servlet |
| 7 | SDK Web Server returns the Servlet's response to the Web Browser. | HTTP |
| 8 | Web Browser displays the HTML Page (requests the Applet if necessary) | HTTP |
| 11 | Web browser displays Client Interface Applet or activates the plug-in,<br>The arguments are the data to sign and possibly a URL | Browser |
| 12 | User clicks button in to approve signing of form data. | GUI |
| 13 | Client Interface (applet or plugin) calls Smart Card API to request that the Smart Card sign an SHA-1 hash of the form data. | Client Interface |
| 15 | User enters PIN when driver ask for it. | OS Dialog |
| 18 | Smart Card API returns signed form data to Client Interface. | Client Interface |
| 19 | Client Interface makes a HTTP connection to the SD1( Web Server and submits the signed form data. | HTTP |
| 20 | SDK Web Server passes request to Servlet | Servlet |
| 21 | Servlet passes request to Filter Engine. | RMI |
| 22 | Filter Engine calls Bank Interface with signed data. | RMI |

Fig. 13A

| | | |
|---|---|---|
| 23 | The Bank Interface calls the Open Card API to request that the HSM sign an SHA-1 hash of the request to the bank. | Java Function Call |
| 24 | Open Card API calls HSM OS Driver | Java Native Call |
| 25 | HSM OS Driver calls HSM to perform signature. | OS-Level Hardware Call |
| 26 | HSM OS Driver returns signed request to Open Card API | Java Native Call |
| 27 | Open Card API returns signed request to Bank Interface | Java Function Call |
| 28 | Bank Interface calls the relying party's bank. | Warranty/Status Check |
| 29 | Relying party's bank calls the issuing party's bank. | Warranty/OCSP |
| 30 | Issuing party's bank returns a signed response to the relying party's bank. | Warranty/OCSP |
| 31 | Relying party's bank then calls the root. | Warranty/OCSP |
| 32 | Root returns a signed response to the relying party's bank. | Warranty/OCSP |
| 33 | Relying party's bank returns a signed response to the Bank Interface. | Warranty/Status Check |
| 34 | Bank Interface validates the signed data and then records the transaction in the log. | File I/O |
| 35 | Bank Interface validates the signed data and then stores the signed data and the signed response from the relying party's bank into the SDK's database. | JDBC |
| 36 | Bank Interface returns an OK or failure result to Filger Engine | RMI |
| 37 | Filter Engine returns failure result to Servlet or passes on initial request to App Server. | RMI |
| 38 | Servlet builds response indicating failure for SDK Web Server. | Servlet |
| 39 | SDK Web Server returns servlet response to the browser if failure. | HTTP |
| 45 | Web Application's Web Server calls the Web Application | ISA |
| 46 | Web Application generates and returns its response. | ISA |
| 47 | Web Application's Web Server returns the response to the Filter Engine | HTTP |

Fig. 13B

| | | |
|---|---|---|
| 48 | Filter Engine returns response to Servlet. | RMI |
| 49 | Servlet returns response to SDK Web Server | Servlet |
| 50 | SDK Web Server returns response to Web Browser | HTTP |

Fig. 13C

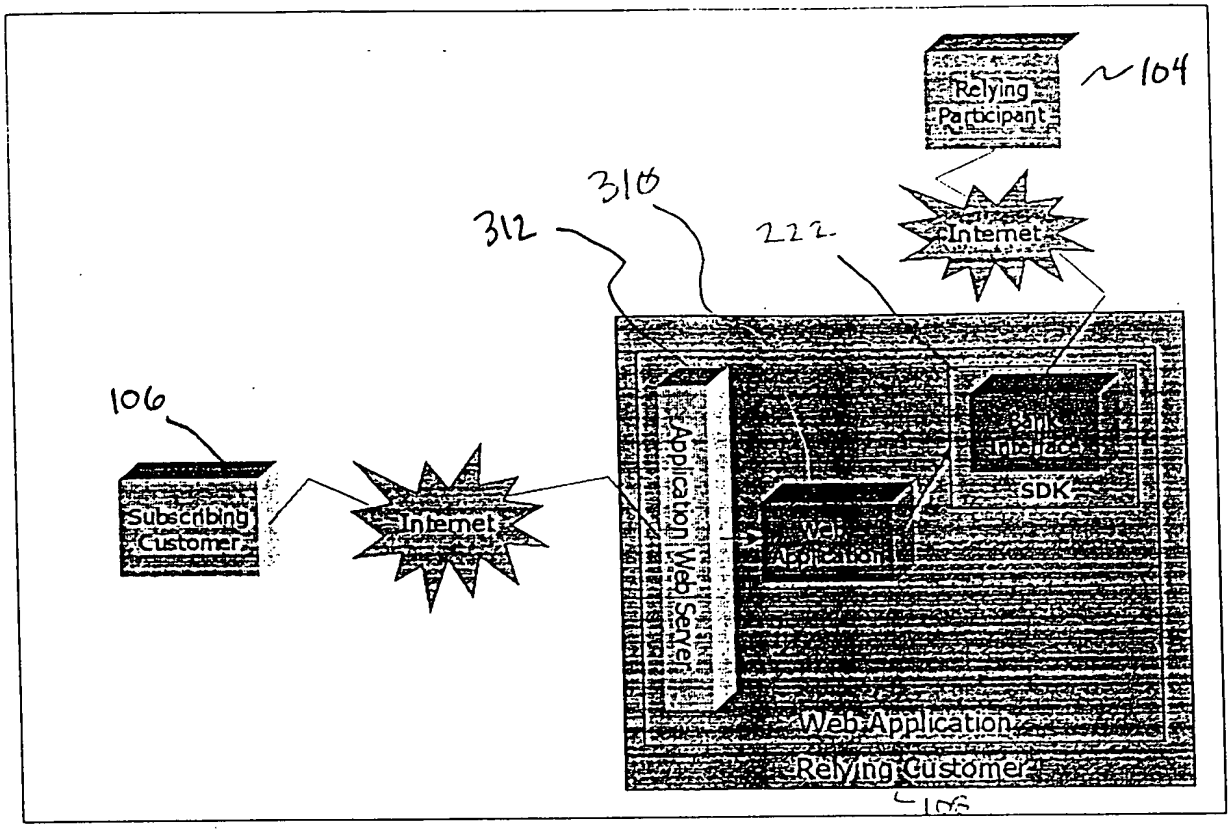


Figure 14 DSMS Active Integration

| # | Description | Protocol |
|---|---|---|
| 1 | User requests form that will require signing when submitted. | HTML UI |
| 2 | Web Browser sends request to Web Server. | HTTP |
| 3 | Web server forwards request to Web Application. | ISA |
| 4 | Web Application returns an HTML page for the web server to return which references the Client Interface | ISA |
| 5 | Web Server returns the HTML Page to Web Browser. | HTTP |
| 6 | Web Browser requests Client Interface from Web Server. | HTTP |
| 7 | Web Server retrieves Client Interface. | OS File System |
| 8 | Web Server returns Client Interface. | HTTP |
| 9 | User clicks the submit and sign button in the web page. | HTML UI |
| 10 | Web Browser calls Client Interface. | Client Interface Technology |
| 11 | Client Interface calls Windows PC/SC to have Smart Card sign data. | OS API |
| 12 | User enters PIN. | OS Dialog |
| 13 | Windows PC/SC calls Smart Card to sign data. | OS-Level Hardware Call |
| 14 | Windows PC/SC returns signed data to Client Interface | OS API |
| 15 | Client Interface returns signed data. | Client Interface Technology |
| 16 | Web Browser posts signed data. | HTTP |
| 17 | Web server passes signed posting to Web Application. | ISA |
| 18 | Integration Code added to the Web Application calls the Bank Interface to verify the signature on the form. | Bank Interface Technology |
| 19 | Bank Interface calls HSM OS Driver to sign request. | OS-API |
| 20 | HSM OS Driver calls HSM to sign request. | OS-Level Hardware Call |

Fig. 15A

| | | |
|---|---|---|
| 21 | HSM OS Driver returns signed request to Bank Interface | OS-API |
| 22 | Bank Interface calls the Relying Party's Bank. | Warranty/Status |
| 23 | Relying Party's Bank calls the Issuing Party's Bank. | Warranty/OCSP |
| 24 | Issuing Party's Bank returns a signed response to the Relying Party's Bank. | Warranty/OCSP |
| 25 | Relying Party's Bank calls the Root. | Warranty/OCSP |
| 26 | Root returns signed response to Relying Party's Bank | Warranty/OCSP |
| 27 | Relying Party's Bank returns signed response to the Bank | Warrant/Status |
| 28 | Bank Interface stores the signed data and the signed OK response from the relying party's bank into the Signed Documents repository. | Database-Access API |
| 29 | Bank Interface writes transaction log message. | File I/O |
| 30 | Bank Interface returns result to Web Application. | Bank Interface Technology |
| 31 | Web Application interprets the form post and returns the next page to the Web Server or an error. | ISA |
| 32 | Web Server returns the page to the Web Browser. | HTTP |

I

Fig. 15B

```
# Sample Servlet Properties File

# The name of the RMI Server class file
Servlet.fename=com.root.filterengine.RmiFEServer
# The IP Address/URL of the Filter Engine's RMI Server
Servlet.feip=rmi://10.1.20.163
```

Fig. 16

```
# Sample Filter Engine Properties File

# The name of the RMI Server class file
SERVER_NAME=com.root.filterengine.RmiFEServer

# The IP Address/URL of the Filter Engine RMI Server
IP_ADDRESS=10.1.20.163

# Location of the Policy file
POLICY_FILE=/dev/src/filterengine/policy

# URL of the Web Application
WEB_APP_URL=http://root8.root.com

# Location of the Rules file
RULES_FILE=/root/rules/RulesFile.txt

# The Name and Location of the Log File
LOG_PATH-/root/logs/fe

# Need'/' at end of CLASS_DIR
CLASS_DIR=file:/root/

# The name of the RMI Server class file
SSL_CERTS_DIRECTORY=/root/sslcerts
```

Fig. 17

```
# Sample Bank Interface Properties File -- Log file Path & type (Flat/DB)
LOG_PATH=/root/logs/bi
LOG_TYPE=TRANS_DB
#LOG_TYPE=TRANS_FILE

# Cryptographic Engine Type/Log File
#HSM_TYPE=com.root.hsm.HSMCryptoPKCS11
HSM_TYPE=com.root.hsm.HSMCryptoSoftware
HSM_LOG_FILE=/root/logs/hsm.log

# Relying Customer Certificate/Private Key for Software Engine
REQUEST_CERT=/root/certs/RelyingCustomer.crt
REQUEST_PRIV_KEY=/root/certs/PrivateKey.pvk

# Root Certificate for Software Engine
ROOT_CERT=/root/certs/Root.crt

# Specify the RMI Server Name/IP address
SERVER_NAME=com.root.bankinterface,RmiBIServer
IP_ADDRESS=root11.root.com

# BankInterface policy file
POLICY_FILE=/root/policies/bipolicy

# CSP Responder URL and SSL
RESPONDER_URL=https://testlab8.qa.valicert.com:90/

# Need '/' at end of CLASS_DIR
CLASS_DIR=file:/root/
#IssuingBank Certificate.  Only for test until we get new SmartCards
DEBUG=/root/certs/IssuingBank.crt

#Directory of trusted CA Certificates for SSL.
SSL_CERTS_DIRECTORY=/root/sslcerts/
```

Fig. 18